Employing Human Knowledge to Solve Integrated Coordination Problems

Wei Chen, Kaizhi Tang, David Mihalcik, Yunshen Tang
Intelligent Automation, Inc.
{wchen, dmihalcik, ytang, ktang}@i-a-i.com

Edmund Durfee
University of Michigan
durfee@umich.edu

Melanie Dumas
DARPA / IPTO
Melanie.Dumas@darpa.mil

ABSTRACT

An Integrated Coordination Problem involves solving multiple related subproblems that collectively satisfy the requirements of a user, including subproblems that depend on the user's participation to solve. Fundamental challenges in solving such a problem include defining mechanisms to solve the individual subproblems, formulating the information and control flow between these mechanisms that supports flexible end-to-end problem-solving, and providing access for people to oversee and participate in the problem-solving process. In this paper, we describe a multi-agent architecture that addresses these challenges by embodying mechanisms in computational agents and by treating the collective problem-solving across agents and people as a collaborative process. We argue that our approach exploits concepts that straddle the boundary between collaborative technologies and multi-agent systems, and demonstrate its advantages and capabilities in the context of an emergency medical response scenario.

KEY WORDS: Architectures and Design of Collaborative Systems, Intelligent & Autonomous Agents in Collaboration, Multi Agent Systems in Collaboration

1. INTRODUCTION

Multi-Agent Systems (MAS) has grown into an interdisciplinary field that embraces many previously distinct research areas, but continues to face challenges of scalability and real-world problems (see, for example, the question raised by Hendler of "where are all the intelligent agents?" [1]). Particularly, MAS/coordination research investigates the underlying algorithms and mechanisms that allow intelligent agents to work with each other, and also possibly with people, to achieve high-level goals that are beyond their individual capabilities. However, it has not embodied adequate considerations from the human users' point of view, which falls into the strength of the currently separate, yet highly related field of Collaborative Technologies and Systems (CTS). CTS investigate the design and development of effective environments or tools that help *human users* work together in a distributed collaborative, possibly virtual, fashion. It is notable that MAS/coordination and CTS share a common driving question about how multiple entities – intelligent agents and/or humans– work together to carry out related tasks to jointly solve problems. Thus, it is natural to combine the strengths of MAS and CTS to address complex real-world problems. One particular approach we describe in this paper is human-agent assisted human-to-human activity collaboration, which is motivated by real world problems (i.e., emergency medical scenarios), and is oriented toward research issues (distributed collaborative problem solving) that reveal synergies between CTS and MAS.

This paper explores the synergies between these research areas by detailing the design and development of a solution to the integrated coordination problem involving humans and computational agents. This paper describes how we have elaborated and implemented our previously-reported conceptual ideas about human agent collaboration (HAC), as applied to a simulated combat medical scenario [7]. Specifically, we describe the overall system architecture, information flow and control flow, distributed planning and scheduling, and an implementation of a human-centered system integration (HSI) scheme to facilitate the HAC processes. We demonstrate the HAC system in a specific scenario of emergency medical response. Our HAC framework solves the integrated coordination problem by combining and controlling interactions between mechanisms for achieving the various previously separate steps of complex distributed collaborative problem solving (e.g., knowledge elicitation, problem specification and analysis, planning and scheduling (matchmaking), plan/schedule update, HAC update, etc.).

This paper is organized as follows. Section 2 describes related efforts in the fields of human-agent collaboration and human-to-human activity coordination. Section 3 briefly summarizes our previously described conceptual framework for human-agent collaboration. Section 4 explores how human expertise could/should be incorporated in complex distributed collaborative problem solving. Section 5 presents the technical details about the design and implementation of our human-agent collaboration architecture as a software system, including its major functional components. Finally, we summarize our results and discuss our future research directions.

2. RELATED WORK

Human-agent collaboration to support cooperative problem solving by people is inspired by, and builds upon, several lineages of research. Computer-supported cooperative work [3], for example, has over time developed increasingly sophisticated computational infrastructures for helping people work together, ranging from concepts as simple as tools to help people jointly edit documents, to as complex as so-called collaboratories that streamline joint research and discovery in various scientific fields [17]. Generally, such collaborative technologies provide the infrastructure for propagating the impacts of peoples' decisions among participants to ensure critical joint awareness and coordination.

Agent technologies emphasize decision-making for computational entities themselves, where such agents are tasked with making decisions on behalf of their users in situations that are dull, complex, fast-paced, or dangerous. The emphasis in agent research has traditionally been in endowing agents with the "intelligence" to act autonomously (without human intervention), and to coordinate autonomously with other agents to collectively accomplish goals as a multi-agent system [8][12][17].

The boundary area between these fields has, however, been explored to some extent, from both directions: collaborative technologies that can proactively make routine decisions and agent technologies where agents help people work together. Indeed, a growing emphasis is in the area of cognitive assistance, where an agent is closely paired with a person (or group of people) to help manage the person's activities and coordinate activities across people. Examples of such systems include Electric Elves [5], EPCA/CALO [2], and Coordinators [13][18].

Our project is in a similar spirit, whose goal is to coordinate peoples' activities. Our work also, however, draws on ideas of human-agent collaboration, where a person and his/her agents work together to combine their expertise to solve complex problems in a mixed-initiative manner (where the human and agent each take initiative to move problem solving forward) [7]. In our work, we draw on all of these ideas to develop agents that are powerful problem solvers in their own right, but which are able to accept (and actively seek) help from people in the course of problem solving specifically to solve problems of constructing and managing teams of people who are themselves cooperatively solving problems in a domain such as emergency combat medicine.

3. PREVIOUS WORK

In [7], we presented an initial design of and a conceptual solution to the integrated coordination problem for employing human knowledge within human-agent collaboration processes in a simulated combat medical scenario. The combat medical scenario represents a real-world problem requiring the location and teaming of human expertise in an on-demand fashion, and our work outlined the design of a human-agent collaboration (HAC) framework for solving this problem. We proposed the major technical components of the HAC framework (including a representation of HAC system resources, a representation of tasks and environments – extended hierarchical task networks (EHTNs) [3], and an associated pre-planning toolkit for the EHTNs), and introduced our ideas for the technical steps necessary to carry out the HAC process (including team formation, task decomposition and allocation, negotiated processes facilitating HAC, etc.).

That early paper focused on the clarification of the problem domain and the formulation of the core research issues. Although schematic, this previous effort paved the way for the subsequent technical design and development of the functional HAC software system described in this paper, e.g., the information and control flow among the proposed technical components, a brief introduction to the agent communication language (ACL) and communication protocols to use for the message transfer, the interfaces for a human actor in different roles (e.g., a user of the HAC system; a human expert – as a resource – who will be invited to join a capable team to participate in an HAC problem solving process; a domain expert who specifies, possibly EHTN-based, task structures) interacting with the HAC system, and the actual software development issues (e.g., the data storage and management scheme), which will be discussed in detail in this paper.

4. HUMAN ROLES IN HAC

Human-Agent Collaboration embraces the complementary strengths of humans and computational agents as problem solvers. As evidenced in the kinds of technologies we are building in this project, our emphasis in developing computational agents is in exploiting their abilities to keep track of vast amounts of information (a database of experts, complex calendar information, an ontology for various roles and their relationships, timing and interaction constraints) and to quickly examine large problem spaces (assignments of experts to roles, propagation of timing relationships among activities, optimization of costs, etc.) to rapidly filter out infeasible options for expert teams.

However, since the experts are (at least often) humans, a variety of constraints and preferences that have to do with (sometimes irrational) human nature might prove difficult to express and use well and often even harder to acquire in the first place [19]. People can have biases and preferences about with whom they interact and what they like to do at particular times of the day that they might be reluctant to articulate due to embarrassment, timidity, or even fear of retribution. Yet, solutions to expert teaming problems that fail to respect these are doomed, as people will find excuses to abandon such teams. Thus, humans should be engaged in the teaming process to steer it towards realistic solutions.

The view we adopt in our work is to assume that people will have preferences and constraints that will be unstated explicitly, but will be indirectly revealed upon an opportunity to provide feedback on partial and/or tentative teaming solutions that the agents identify as satisfying the articulated parameters of the problem. By expressing preferences over a handful of proposed solutions, or pointing out components that need changing in a proposed solution, for example, people can contribute to the problem-solving process to formulate better solutions than the computational agents can do alone. It is in this spirit that we have been developing our HAC techniques.

5. THE HUMAN AGENT COLLABORATION (HAC) SYSTEM

Before introducing our HAC system, we have slightly changed our application domain: instead of emergency combat medicine, we ground our work in a civilian medical emergency scenario due to a greater availability and accessibility of data. A civilian medical emergency requires similar types of resources (e.g., doctors, nurses), but can differ in the manner in which experts will participate in HAC. For example, a civilian medical expert has more latitude in responding to an HAC request: a civilian doctor may turn down an HAC request without elaborating the reason while a military doctor needs to follow orders. In our system, this kind of difference is technically trivial thanks to our uniform specification of experts' profiles (e.g., capabilities and personal schedules).

The input to the HAC system is a medical emergency case that requires quick response and proper treatment from a high-quality team, whose members may be dynamically found from different medical sites. It is the HAC's responsibility to assist its users with different levels of medical expertise (e.g., an EMT (emergency medical technician) with good medical expertise, a firefighter with only basic medical knowledge, or even a regular guy passing by reporting this emergency) to evaluate the case, suggest suitable response steps, form a qualified team to carry out the steps, determine the ordering and timing of the steps to generate an agenda, and finally execute and possibly revise the agenda in response to dynamics, uncertainties, and contingencies.

The corresponding research issues of HAC are complicated and consist of many traditionally separate subproblems, many of which are theoretically and/or practically intractable by themselves in terms of time and space complexity, e.g., scheduling. Existing solutions in the literature to the separate component subproblems cannot be simply combined to form an overall solution to this complex problem; rather this is an *integrated coordination problem*, and demands a carefully integrated solution.

We define an **integrated coordination problem** as the problem of managing *a complex distributed collaborative problem solving process among distinct constituent components by properly interconnecting the components and suitably managing their information and control flows in ways that lead to successful and efficient collaborative problem solving*. Figure 1 is a notional depiction of how the constituent subproblems can interact in the HAC integrated coordination problem. The boxes specify the individual subproblems. The arrows represent the information flow and control flow. In our emergency medical scenario, the ***problem specification and analysis*** subproblem is to elicit from the HAC user the needed injury information and evaluate the nature and severity of the injury. The ***planning*** subproblem is to find a suitable course of actions (COA) to provide medical treatment based on the specification and analyzed severity of the injury. The planning subproblem itself is comprised of the ***team formation*** problem which is to identify a team of capable experts, and the ***task allocation*** problem which is to decide which expert should carry out which particular action. The next challenge is to decide when the planned actions should be carried out, accounting for when the experts are available, and this is the ***scheduling*** subproblem. After the temporal information is specified, the result becomes an actionable agenda ready for execution; the ***execution*** subproblem is to ensure that the agenda is carried out, including responding to contingencies that arise. The execution outcomes, whether exactly as predicted or significantly different, should be monitored and analyzed, and the ***result analysis*** subproblem is to interpret the agenda's performance to identify possible improvements/updates in knowledge to improve the solutions generated for other subproblems. Finally, given that the resources are distributed, the **coordination** subproblem is to suitably synchronize and share agents' local views to ensure coordinated behavior, such as that experts involved in an online joint consultation

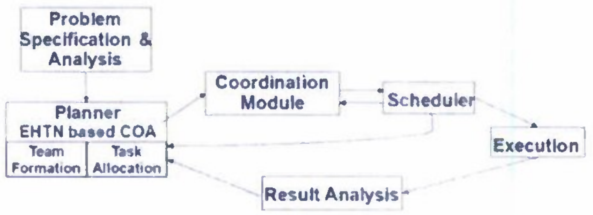

Figure 1. The Integrated Coordination Problem.

join the consultation at the same time. As explained above, solving the overall integrated coordination problem is extremely difficult, and thus in this paper we focus on this overall coordination problem without delving into domain-dependent details of medical actions and plans.

5.1. HAC System Architecture

Now that we have described the overarching integrated coordination problem and its component subproblems, we turn to the specific software architecture that we have developed for solving the subproblems and for controlling the information and control flows between them. Our architecture employs a multi-agent-system-based design and implementation strategy to create an HAC software system. The HAC software system includes various constituent functional components, implemented as agents (e.g., a case manager, a core HAC agent, a matchmaker, and a scheduler), the information flow and control flow among these agents, and the underlying algorithms and mechanisms realizing the HAC capabilities.

A schematic of our HAC architecture is in **Figure 2**. The boxes represent the reasoning agents. The arrows represent the information and control flow in response to a user's problem-solving request – referred to as an HAC case. The numbers with the arrows indicate the order of steps during a typical HAC process without any exceptions arising.

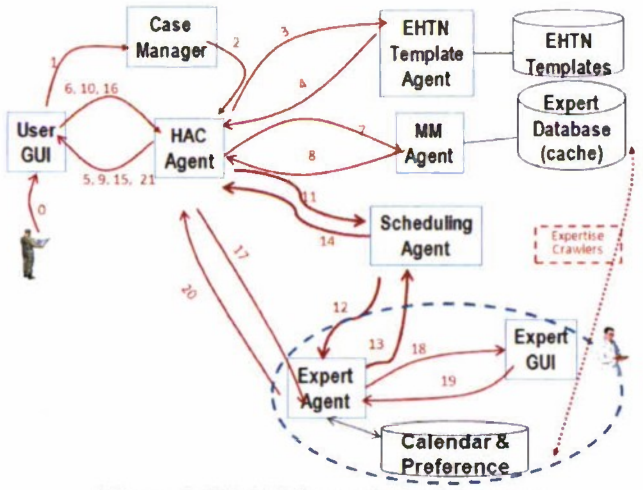

Figure 2. HAC Schematic Architecture.

The component capabilities are designed and implemented as intelligent agents, and a typical HAC process is briefly explained as follows. An HAC user submits a new medical case to, and carries out subsequent interaction with, the HAC through the *User GUI* agent. The *Case Manager* receives the input and collects the case features (case ID, user profile, patient medical condition, etc.) and then turns control over to the HAC agent. The *HAC Agent* is the core of the HAC system and orchestrates the system functionalities by finding the best response plan from the *EHTN Template Agent*, then finding candidate experts (from potentially tens of thousands of people with relevant expertise) to form the a satisfactorily capable team via the *Matchmaking (MM) Agent*, and then setting up the ordering and timing of the steps by employing the *Scheduler Agent*. The EHTN Template database agent stores various regulated medical response procedures represented in the form of extended hierarchical task networks (EHTNs) and provides the most suitable task structure in response to a given case. The *Calendar and Preference database agent* stores experts' personal scheduling information and biases, if an expert chooses to provide such information via his/her *Expert GUI agent*. The *Expert Database agent* provides a cache of the Calendar & Preference Database for the Matchmaking agent to compute the best team formations with updated information. Steps 21 and 22 indicate confirmation step to the experts – not a dead end. The dashed arrow represents the periodic update of the cache data by automatic Expertise Crawlers (details omitted because they are outside the scope of this paper). Overall, **Figure 2** summarizes our HAC solution containing both computational agents and humans. The dashed oval part indicates the generation and management of *an asset "Internet"* to locate human expertise on demand for distributed collaborative problem solving.

5.2. Information Flow and Control Flow

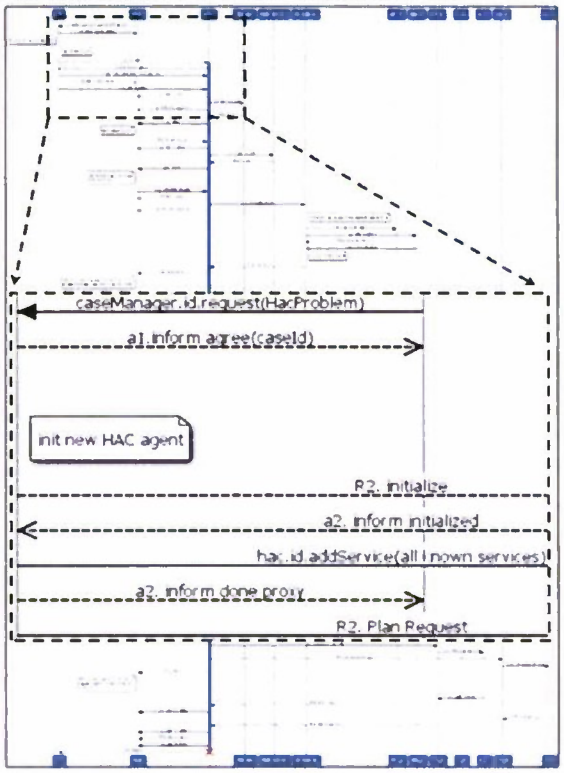
caseManager.id.request(HacProblem)
a1.inform agree(caseId)
init new HAC agent
R2. initialize
a2. inform initialized
hac.id.addService(all known services)
a2. inform done proxy
R2. Plan Request

Figure 3. Communication Flow Diagram Among HAC Components For Regulating HAC Updates.

Figure 3 provides an admittedly difficult-to-read screenshot of message exchanges among the constituent

HAC agents. A small portion of the exchanges for HAC case initialization between an HAC user, the Case Manager agent, and the HAC agent is magnified, and will be further clarified in **Figure 4**. The HAC message exchanges are a realization of the FIPA ACL (Foundation for Intelligent Physical Agents, Agent Communication Language) [13] to achieve interoperability with external systems.

One of the most important features of this information flow and control flow is the integration of humans in the process. A human may act as either a user who submits a medical case to the HAC system, or a domain expert who is invited to join a team to provide her medical expertise to the case, or a system engineer in a domain who specifies the suitable task structures (represented using EHTNs) for various cases. Computationally, there is no need for explicit distinction between humans and software agents, because GUI agents may act as humans' delegates in the HAC system; logically, humans' responses – not necessarily modeled with underlying reasons – are incorporated on the fly during the HAC process. Human expertise provides guidance to the HAC system to achieve suitable solutions quickly by filtering out large fractions of infeasible solution spaces and at the same time incorporating humans' biases.

5.3 Functional Components

In what follows, we summarize the component agents in the HAC system, pointing out how their functionality is achieved, and how they interact with people and other agents as part of the overall HAC process.

5.3.1 User GUI and Agent

The User GUI agent is the users' delegate to the HAC system. A user can set up the level of interaction based on her level of expertise or preference. An experienced user (e.g., an EMT requesting follow-up emergency treatment for a traffic accident victim) may wish to monitor/confirm every HAC step following the message flow in Figure 2. An inexperienced user (e.g., a helpful bystander who witnessed the accident) does not have adequate expertise and thus relies on the HAC system in a largely *automated* mode to guide the user through the response processes. In a fully automated mode, the HAC steps will proceed until reaching an agenda without any human intervention.

The User GUI screen shots are omitted due to space, but we adopted a practical design for HAC users in the form of Wizard Dialogs. A ***Wizard Dialog*** [10] box is constructed from a number of panels, and each panel contains user-configurable components such as radio buttons, sliders, text fields, etc., for every HAC reasoning process. The idea is that a user of the HAC system, by pressing either the `Next` or `Back` buttons, can "flip" across these panels, entering information on each one until she completes the entire HAC procedure from the initial problem specification to the final step of agenda generation.

Given an HAC user needs to set preconditions and constraints to a particular reasoning step and takes actions (e.g., confirm, select from, or ask-for-more candidate solutions at each step) on the results from the previous step, it is natural to adopt the Wizard-Dialog-style interactions for human users. Six major steps for HAC have been designed as sequentially dependent, integrated procedures:

- User Profile – setting up a user's general preferences or utility functions that guide the underlying behaviors of the HAC reasoning;
- Patient Information and Medical Record Form – the user specifies patient information and domain-dependent information (e.g., a medical emergency condition);
- Task Structure – the user will have candidate task structures, a.k.a., courses of actions, as results from the analysis of a medical case in the previous step, and may confirm, or choose from, these candidate task structures for the next step, and can also update the information embedded in a candidate task structure for further processing; additionally, the user can specify certain conditions/constraints on the next step of matchmaking;
- Matchmaking (team formation) – the user will face possibly multiple candidate team formations based on resource analysis, will be able to confirm/choose from the teams for the next step, and can specify conditions/ constraints on the next step of scheduling;
- Scheduling – the user will face possibly multiple candidate schedules for each of the results from the previous step, may confirm/choose from the resulting candidate schedules, and can specify certain conditions/constraints on the next step if applicable;
- Agenda – The user will confirm with the assets (including human experts) that will be ready to perform the scheduled tasks. An agenda is a confirmed schedule.

With this implementation, an HAC user may choose to sequentially and manually go through these steps (generally for experienced users) or may set a suitable user profile for automatic HAC processing (for inexperienced users or for time-critical situations), or can participate in only parts of the process, where the user has knowledge or insights to contribute that will improve the efficiency and/or outcome of the HAC process.

5.3.2 Case Manager

As shown in Figure 4, the Case Manager acts as a broker between user agents and HAC agents. To initialize a conversation, or to look up an active HAC agent, a user agent submits a request to the system's single Case Manager, which is in turn responsible for finding or creating an HAC agent that is dedicated to that particular case. Due to the potential for high demand for its attention, the case manager service is kept as simple (and thus fast) as possible. In our current implementation, it performs HAC agent initialization, provides directory services, and logs its activities; these may be separated into different agents in the future.

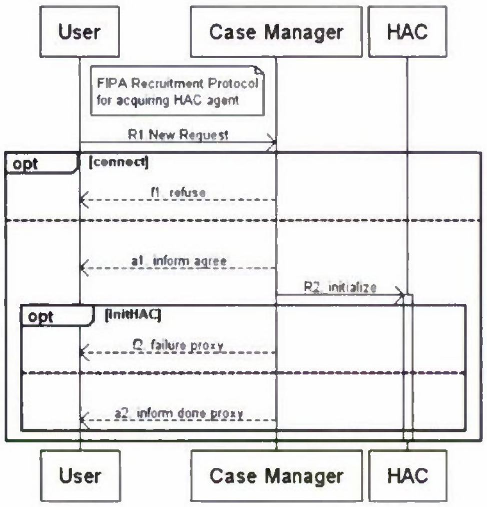

Figure 4. Comm. Among User, HAC, & Case Manager.

5.3.3 HAC Agent

The HAC Agent is the point of contact and arbiter of all interactions relating to the content of the user's request and its solution process. The HAC Agent communicates with several service agents to get a set of possible plans, agendas, and team assignments, possibly with interaction from other agents who may have information to help address the request. The HAC agent enlists the identified experts for the scheduled activities, confirming their willingness and availability. The HAC agent collects update information on the execution of the activities and tasks, and records the success or failure of task executions.

Each HAC agent communicates with three service agents to build, verify, and clarify the solution to a user's task request. The EHTN Template agents refine requested tasks (using our EHTN model) into sequences of actions with associated constraints. The Matchmaker (a/k/a the matchmaking agent) assigns resources or experts to the '*Position*' parameters of a request, as will be explained shortly. The Scheduler agent specifies temporal information that details when the planned actions take place, ensuring an effective and synchronized timing of joint actions (e.g., a joint consultation among multiple experts) at times when those experts are available. After the problem-solving process reaches agenda stage, the Expert agents, who track the schedules of their respective human experts and act as the user interface to their experts, help to revise a problem, potentially reflecting feedback from their associated humans about proposed agendas and teams. Similarly, this process of involving domain experts will be moderated by the HAC agent for status monitoring and coordination purposes.

5.3.4 EHTN Template Agent

In our previous paper [7], we discussed the advantages of employing an expressive task/environment representation, extended hierarchical task networks (EHTNs), as a solution to the knowledge representation problem for HAC. Notably, one case may have multiple corresponding EHTN structures and the HAC user has a chance to select either only one or pass all the candidate structures to the subsequent HAC processes for consideration.

We will not delve into detailed discussion here, but simply state that an advanced reasoning solution, which is able to carry out pre-planning, with the flexibility for human users to deal with dynamics, uncertainties, and contingencies on the fly, has the advantages of eliminating human actors' *ad hoc* mistakes, providing guidance during complex hard-to-memorize response procedures to human users (especially inexperienced users), and still maintaining effective control over task planning and executions [7] [3].

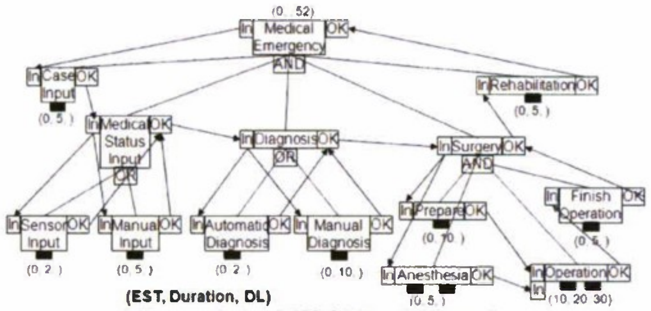

Figure 6. An EHTN Task Template

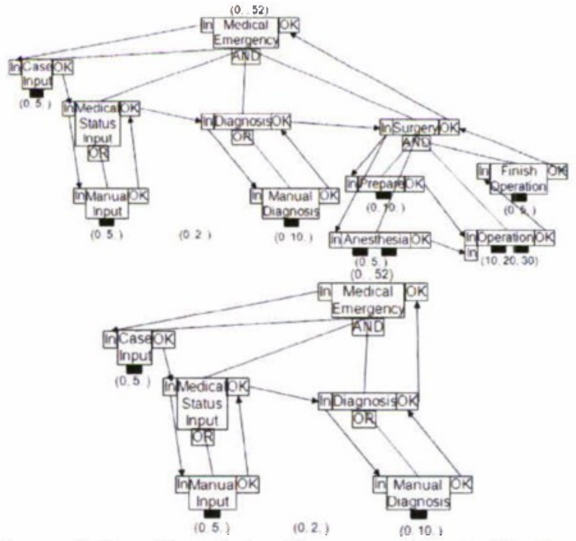

Figure 5. Two Templates Corresponding to Medium and Low Levels of Severity.

The pre-planning function generates task structure *templates*, defined as regulated response procedures for various cases that share common features within the same category. A sample task template in response to a medical emergency case is shown in **Figure 6**. For example, the response procedures for treating leg trauma with the same level of severity are the same no matter whether the injury is in the left or in the right leg or who the particular victim is. Thus, the procedure is always as shown in **Figure 6**: (1) case input – potentially a 911 operator receives and generates an emergency case, (2) medical status input – the injury is inspected and described, (3) diagnosis – an initial assessment of the injury is performed; (4) surgery – if necessary, the actual surgical treatment is prescribed, which itself consists of sub-tasks, and finally (5) rehabilitation– if applicable, therapy and other follow up treatment to recover from the injury are performed. However, the severity and the type of an injury may result in different treatment procedures. For example, a back trauma may result in a different response procedure than a leg injury; or a simple scratch on the leg will result in radically simpler (and less invasive) treatment. Figure 5 shows two different task templates corresponding to medium and low levels of severity.

5.3.5 Matchmaking Agent

Notably, one EHTN task structure may have multiple corresponding candidate teams that meet a user's objective function (e.g., the most capable team). The user has an opportunity to choose either one or pass all resulting teams to subsequent HAC processes. We only address a single matchmaking execution next.

A matchmaking agent will be responsible for assigning a set of positions that are grouped by several actions to a set of available human experts to satisfy the expertise requirements of the positions and respect the preference profiles of the experts at the same time. One or more *position* with role specifications are placeholders associated with an action. The position definition introduces the concept of concurrency to task structure – the positions associated with a common action node overlap in time. For example, a surgery action contains two positions: surgical operation (by a surgeon role) and support operation (by a surgical nurse role) that should be assigned to a surgeon and a nurse respectively. This assignment problem can be modeled as an integer programming problem and solved by a standard mathematical programming package.

We describe this problem with a small example as shown in Figure 6. Suppose that an HAC problem has n positions (the black boxes) that are associated with m actions (the bottom level nodes). The membership matrix $\{a_{ij}\}$ denotes the association between the actions and positions, where

$$a_{ij} = \begin{cases} 1, & \textit{if position i belongs to action j} \\ 0 & \textit{otherwise} \end{cases}$$
$$\forall i \in [1,n], j \in [1,m]$$

It is required that each position can only be assigned to one action, such that

$$\sum_{j=1}^{m} a_{ij} = 1 \qquad \forall i \in [1,n]$$

Suppose that there are p roles and each position has a specific required role. Let the matrix $\{b_{ij}\}$ denote the association relationship between positions and roles, where

$$b_{ij} = \begin{cases} 1, & \textit{if position i requests role j} \\ 0 & \textit{otherwise} \end{cases} \qquad \forall i \in [1,n], j \in [1,p]$$

It is required that each position can request only one role, such that

$$\sum_{j=1}^{p} b_{ij} = 1 \qquad \forall i \in [1,n]$$

Suppose that we have o experts. Each expert has a capability profile, preference profile, and availability profile. The *capability profile* denotes the expert's level of expertise for each of the roles. Let the matrix $\{c_{ij}\}$ denote the capability profile, where c_{ij} is a number normalized from 0 to 10, denoting the expertise level of expert i for role j. For $c_{ij} = 0$, expert i definitely has no expertise for role j, and should never be assigned to play that role. The *preference profile* is similar to the capability profile, denoting the preference level of an expert for each of the possible roles. Let the matrix $\{r_{ij}\}$ denote the preference profile, where r_{ij} is a number from 0 from 10, denoting the preference level of expert i for role j. For $r_{ij} = 0$, expert i definitely does not want to perform the role j. Clearly, the matrix $\{r_{ij}\}$ is correlated with the matrix $\{c_{ij}\}$ in that an expert who is incapable of performing a role should want to avoid it: $c_{ij} = 0 \Rightarrow r_{ij} = 0$.

The *availability profile* is a comprehensive metric on the remaining free time of an expert over the scheduling period. To reduce the complexity of solving the planning and scheduling problem in HAC, we decompose the problem into two sequential problems: a matchmaking problem and a scheduling problem. The scheduling problem has hard constraints on the time slots to assign tasks (that is, the positions). However, we first are solving the matchmaking problem, without considering the experts' scheduling constraints. This can easily make the scheduling problem infeasible. To reduce the probability of schedule infeasibility, we introduce the availability profile. We require that an availability metric should reflect the amount of remaining unscheduled time for an expert over the future scheduling period and the number of positions for which he or she is already scheduled. Note that the availability profile is not exact information for the availability of the expert, but rather a statistical summarization of availability. Otherwise, the matchmaking problem must deal with the scheduling information, which will make the problem difficult to solve.

Let the vector $\{v_i\}$ denote the availability profile where each element is a value between 0 and 1 that denotes the fraction of time during the scheduling period that the expert i is still free.

Since availability is critical, our matchmaking formulates the optimization problem as maximizing the product of availabilities for the assigned experts (roughly corresponding to finding the combination of experts with the highest joint probability of being scheduled, where this uses the simplifying assumption that their schedules are independent) to the positions and at the same time to put thresholds on the capabilities and preferences for the assigned experts.

Let x_{ij} denote the matchmaking results, indicating whether the position i is assigned to expert j, where

$$x_{ij} = \begin{cases} 1, & \textit{if position i is assigned to } \text{exp}\textit{ert } j; \\ 0, & \textit{otherwise.} \end{cases}$$
$$\forall i \in [1,n], j \in [1,o]$$

We formulate the matchmaking problem as an integer programming problem.

$$\max \sum_{i=1}^{n} \sum_{j=1}^{p} \sum_{k=1}^{o} b_{ij} v_k x_{ik}$$

$$s.t. \sum_{k=1}^{o} x_{ik} = 1, \qquad \forall i \in [1,n] \qquad (1)$$

$$\sum_{i=1}^{n} a_{ij} x_{ik} \leq 1, \qquad \forall j \in [1,m], k \in [1,o] \; (2)$$

$$\sum_{j=1}^{p} \sum_{k=1}^{o} b_{ij} c_{kj} x_{ik} \geq \alpha, \; \forall i \in [1,n] \qquad (3)$$

$$\sum_{j=1}^{p} \sum_{k=1}^{o} b_{ij} r_{kj} x_{ik} \geq \beta, \qquad \forall i \in [1,n] \quad (4)$$

$$x_{ik} \in \{0,1\}$$

Constraint (1) requires that every position must be assigned to an expert. Constraint (2) enforces that no expert can act in multiple positions for the same action. Constraint (3) means that the total capability levels of the assigned expert to each position must be above a certain threshold. Constraint (4) implies that the combined preference levels of the assigned expert to each position must be above a certain threshold.

5.3.6 Scheduling Agent

A single team formation may have multiple corresponding candidate schedules that meet a user's objective function (e.g., the earliest deadline). The user has an opportunity to choose either one of these or pass all resulting schedules to subsequent HAC processes, if applicable. We only address a single scheduling execution next.

The scheduling agent assigns time slots (respecting sequential ordering constraints) to a set of positions distributed among several actions that have been assigned to a group of experts as a result of matchmaking. Considering the existing schedules for those experts, a scheduling agent must satisfy the constraints that no additional positions will be assigned to any time slot that has already been assigned. In this paper, since we address medical emergency problems, the goal of a scheduling agent is to attempt to provide emergency treatment as soon as possible. From the perspective of optimization, the above goal suggests the tentative objective of minimizing the summation of the starting times of all the positions. An alternative goal may be to minimize the makespan.

An HAC scheduling problem can also be modeled as a mathematical programming problem using the same techniques as for the matchmaking problem. However, an HAC scheduling problem is much more difficult to model and solve than the matchmaking problem.

First, more decision variables are needed to model the relative sequences between positions, some of which represent the occupied time slots. This adds complexity to the scheduling problem that is similar to the notorious vehicle routing problem [8], where pickup time or delivery time and their sequences are very similar to sequences and starting times in the HAC scheduling problem. Let x_{kij} denote whether expert k has in its schedule that it will perform position i immediately followed by position j, where

$$x_{ijk} = \begin{cases} 1, & \begin{array}{l} \textit{if position i is followed by position} \\ \textit{j in the schedule of } \text{exp}\textit{ert k} \end{array} \\ 0 & \textit{otherwise} \end{cases}$$
$$k \in [1,m], \forall i \in [0,n+1], j \in [0,n+1],$$

Without the matchmaking results, the HAC scheduling problem will be an NP-hard problem and difficult to solve for large-scale problems. With the matchmaking results, the complexity of the HAC scheduling problem has been reduced with suitable approximation into an optimization problem within a constrained local problem space.

Second, continuous variables (starting times) and integer variables (sequences) are mixed together. Such a mixture makes the optimization search process difficult, because the search methods for integer and continuous variables can be very different. The mixture of these two search methods may require more time to converge.

Third, the HAC scheduling problem expresses nonlinearity. To check whether a position (with the associated execution duration) can be inserted before another position, the product of a sequence variable and starting time must be considered. This will cause additional nonlinear constraints. The following equation specifies that if a position follows another position in the schedule of an expert, the starting time of this position must be greater than or equal to the finishing time of the previous position.

$$\sum_{i=0}^{n+1} x_{kij} * (t_i + d_i) \leq t_j \qquad \forall i \in [0, n+1], a_{jk} = 1, j \neq 0$$

As a summary, an HAC scheduling problem is a nonlinear mixed integer programming problem, which is challenging to solve. Duc to the provided matchmaking results, the complexity of the HAC scheduling problem has been reduced significantly.

5.3.7 Expert GUI and Agent

An expert has two main interaction points - a profile view and an agenda view. By editing the profile, the expert can control what kind of requests she receives. Through the agenda view, the expert responds to an HAC request allocated to her, works to design appropriate response plans, and reports the success or failure of tasks.

Expert Profiles

The goal of each expert is to get the most relevant and interesting jobs available. To support that, the expert profile interface allows experts to modify how both the user and the system select them. Most importantly, the profile editor allows an expert to modify his or her capabilities and preferences. This will make the user - and the system - more likely to select the expert for a specific task. Additionally, to avoid unwanted requests, the expert can edit his or her work schedule and preference settings to indicate the likelihood of accepting an HAC request.

Job Matching and Monitoring

While modifying the expert preferences and capabilities will steer appropriate requests to the expert, the major part of the collaboration comes after an expert is offered a task. From here, the expert may agree to the task, refuse the task, or attempt to alter the task. At this point, the selection of a team and agenda from the expert's point of view moves from being a largely computer-dominated task to an interactive, collaborative task. After an expert agrees to a task, the task may be updated with changes to the team composition or other tasks; if the expert's assignment is unchanged, the expert is assumed to still be committed. However, if the expert's task assignment or requested schedule changes due to a conflict or a request by another expert, the expert must re-acknowledge the commitment. Experts may only alter their own tasks; for joint tasks, either expert may request a change of time slot.

5.3.8 Databases and Services

For a complex large-scale distributed collaborative problem-solving system, like HAC, it is imperative to manage its data effectively and efficiently. We target to develop a smart network of heterogeneous human expertise taking advantage of the internet infrastructure. We have designed and developed an Expert GUI that can be executed on heterogeneous devices (e.g., computers/laptops, PDAs) for domain experts to register their expertise and personal schedules - if they want - that will be cached and further synchronized with an HAC Expert (or expertise) database. Expertise information will be inserted either by domain experts via the GUIs manually or by expertise crawlers (not fully implemented yet) automatically. Compared with the large database size (e.g., the number of health care providers in a medium-size city in the U.S. easily exceeds several thousand), a team for a single medical emergency usually requires no more than two dozen medical experts. It is difficult to find the best two dozen team members from the several thousands of candidates in addition to thc feasibility check (whether the task plan can achieve the goal as requested) and the availability check (the team members are all available for action at the time requested).

In responsc to the above challenges, we designed and implemented an underlying Expert database and the corresponding database service agent. A database service agent is a persistent entity that provides results to regulated database queries (e.g., please give me a list of the surgeons with a capability level above N with a flexible schedule between time points T_A and T_B and within a 2 mile range) to support the HAC processes. How to generate and manage the underlying data for HAC is an implementation issue. Our effort ensures data management via reliable services and provides a reasonable problem space for the optimizations described in the previous subsections.

6. CONCLUSION AND FUTURE WORK

We have introduced and implemented a human agent collaboration (HAC) system for facilitating human-to-human activity coordination. The focus of this paper has been the technical details of the software architecture, the information and control, the functionalities of the constituent components, and their associated underlying algorithms. A unique feature of the HAC system is thc integration of human expertise in the problem-solving process, incorporating humans' biases, personal schedules, and the flexibility for manual responses considering thc kind of information not present or not explicitly articulated in real-world scenarios.

On the technical front, one of our current research directions is to explore an even more integrated approach to team formation (matchmaking) and scheduling. We have formulated these two problems within the Hybrid Scheduling Problem (HSP) framework [16]. This allows us to apply state-of-the-art algorithms [3; 10] to simultaneously solving selection and scheduling problems. We plan to exploit our multi-agent architecture to inject a new HSP-based agent into the system which can potentially find schedulable teams of experts faster, and thus seed the more time-consuming schedule optimization process with a team assignment that is known to be feasible, thus reducing backtracking.

Another of our future efforts is to test the applicability of the HAC system in an extended set of domain applications. The HAC is a meta-level problem-solving system independent of domain-specific reasoning. We expect the HAC solution to be applicable to more domains that share the challenge of large-scale distributed collaborative problem solving, e.g., the scenario of cultural expertise on demand that originally motivated this HAC research [7], the combat medical scenario with realistic data, and commercial applications (such as case management in health care, emergency management, and disaster relief). The HAC system has been implemented for users' manual operation. Large scale simulation and experimentation will be carried out in our future work.

ACKNOWLEDGEMENT

The work presented in this paper was partially funded by DARPA through contract# W31P4Q-08-C-0318. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense. Distribution Statement: Approved for Public Release, Distribution Unlimited.

REFERENCES

[1] "The Challenge of Finding Intelligent Agents," *IEEE Intelligent Systems*, vol. 22, no. 4, pp. 3-5, 7, July/Aug. 2007, doi:10.1109/MIS.2007.78

[2] P. Berry *et. al.* "Conflict Negotiation Among Personal Calendar Agents". *AAMAS'06*, pp. 1564–1571, May 2006, Japan.

[3] J. Boerkoel and E. Durfee (2009). "Evaluating Hybrid Constraint Tightening for Scheduling Agents". In *Proc. of AAMAS 2009*, pages 673-680.

[4] P. Carstensen & K. Schmidt. "Computer Supported Cooperative Work: New Challenges to Systems Design", p. 619—636. In K. Itoh (Ed.), *Handbook of Human Factors*, 1999.

[5] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, and M. Tambe (2001). "Electric elves: Applying agent technology to support human organizations." In *Proc. of the Conf. on Industrial Applications of AI*, Seattle, WA.

[6] W. Chen & K. Decker. "Analyzing characteristics of task structures to develop GPGP coordination mechanisms". *5th Intl. Joint Conf. on Autonomous Agent and Multi-Agent Systems*, pages 662-669, Hakodate, Japan, May 2006.

[7] W. Chen, E. Durfee, and M. Dumas. "Human Agent Collaboration in a Simulated Combat Medical Scenario". In *Proc. of the International Symposium on Collaborative Technologies and Systems*, pages 367-375, Baltimore, USA, May 2009.

[8] R. Kohout & K. Erol. "In-time agent-based vehicle routing with a stochastic improvement heuristic". *11th Conf. on Innovative Applications of AI*, 1999. Orlando, FL.

[9] K. Decker and J. Li. "Coordinating mutually exclusive resources using GPGP". The *Journal of Autonomous Agents and Multi-Agent Systems*, 3(2): 133--158, 2000.

[10] R. Eckstein. "Java technical documentation." http://java.sun.com/developer/technicalArticles/GUI/swing/wizard/

[11] L. de Moura and N. Bjørner (2008). "Z3: An efficient SMT solver". In *Proc. Of TACACS-2008*, 337-340.

[12] E. Durfee and T. Montgomery. "Coordination as Distributed Search in a Hierarchical Behavior Space." *IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on DAI, SMC*, 21(6):1363-1378, November 1991.

[13] Foundation for Intelligent Physical Agents (FIPA), Agent Communication specifications. http://www.fipa.org

[14] R. Maheswaran, *et. al.*, "Predictability & Criticality Metrics for Coordination in Complex Environments," *Proc. of the 7th AAMAS, pages 647-654,* Estoril, Portugal, May 12-16, 2008.

[15] G. Olson, A. Zimmerman & N. Bos (Ed.), *et. al.* "Scientific Collaboration on the Internet", MIT Press, November 2008.

[16] P., Schwartz (2007). "Managing Complex Scheduling Problems with Dynamic and Hybrid Constraints". *PhD. Diss., Computer Science and Engin.*, Univ. of Mich., Ann Arbor.

[17] M. Tambe. "Agent architectures for flexible, practical teamwork". *4th National Conf. on AI*, p. 22-28, Providence, 1997.

[18] T. Wagner (2005). "DARPA COORDINATORs". http://www.darpa.mil/ipto/programs/coordinators/

[19] K. Wiegand. "Information theory and human behavior: uncertainty as a fundamental variable in information-processing tasks". DTR, AD0423557, TIC, Oct. 1963.